

November 30, 2016

# Objetivos do Trabalho

O objetivo principal do trabalho foram:

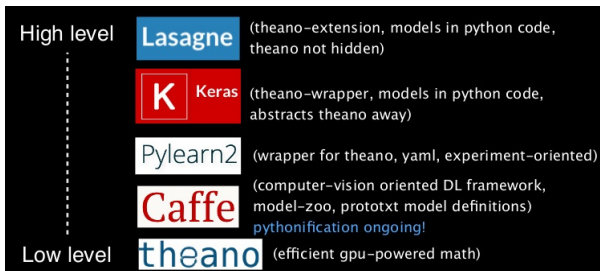
- ▶ Estudar a estrutura e o funcionamento de Redes Neurais Convolucionais 2D
- ▶ Entender como a informação é transformada dentro da rede neural ao avançar nas camadas mais profundas
- ▶ Extrapolar esse conhecimento para redes mais clássicas como a **MLP** estudada durante a disciplina.

# Apresentação do Bruno

# Apresentação do Bruno Canale

# Python - Keras Framework para Machine Learning

- \* Python - Linguagem de programação gratuita
- \* Contém uma quantidade muito grande de Frameworks voltados para *Machine Learning*



Keras foi inicialmente desenvolvido como parte de um projeto de pesquisa chamado de ONEIROS (Open-ended Neuro-Electronic Intelligent Robot Operating System)

## Keras - Exemplo de implementação MLP

```
modelo = Sequential()  
modelo.add(Dense(num-neuronios, init='uniform'))  
modelo.add(Activation('tanh'))  
  
modelo.add(Dense(1, init='uniform'))  
modelo.add(Activation('linear'))  
  
modelo.compile(learning-rate=0.1, optimizer='sgd')  
  
modelo.fit(features-treino, target-treino)  
  
modelo.predict(features-teste)
```

# Base de dados utilizada - MNIST

A base de dados MNIST é composta por 60.000 exemplos de imagens de dígitos em letra cursivas. O dataset é ideal para testes de algoritmos em reconhecimento de padrões por necessitar pouco pré-processamento.

Mais informações no link:

<http://yann.lecun.com/exdb/mnist/>



```
from keras.datasets import mnist
```

```
(X_treino, y_treino), (X_teste, y_teste) = mnist.load_data()
```

## Processamento necessário no atributo target

Um processamento padrão é transformar o conjunto de **atributos targets** em um conjunto de variáveis categóricas. O que seriam variáveis categóricas?

Exemplo: Se a lista de targets é composta por: [1.2, 2, 3, 4.2, 4i] e as classes disponíveis são **real**, **inteiro**, **imaginario**, então uma matriz de transformação seria:

**Table:** Conversão para variáveis categóricas

target	real	inteiro	imaginario
1.2	1	0	0
2	0	1	0
3	0	1	0
4.2	1	0	0
4i	0	0	1



# Implementação da Rede Convolutacional 2D

```
from keras.layers import Activation, Dense, Flatten
from keras.layers.convolutional import Convolution2D
from keras.layers.convolutional import MaxPooling2D

num_filtros = 32
num_conv = 6
num_pool = 4

modelo = Sequential() # instanciar o modelo

modelo.add(Convolution2D(num_filtros, num_conv, num_conv,
                        border_mode='valid',
                        input_shape=(28, 28, 1)))
modelo.add(Activation('relu'))

# adicao da segunda camada convolucional
modelo.add(Convolution2D(num_filtros, num_conv, num_conv))
modelo.add(MaxPooling2D(pool_size=(num_pool, num_pool)))

# camada que transforma ('comprime') a saida em um array 1D
modelo.add(Flatten())

modelo.add(Dense(100, activation='relu'))
modelo.add(Dense(numero_classes_categoricas),
                activation='softmax')
```

# Modelo do Experimento realizado para análise da Rede

- ▶ Após implementação a arquitetura foi testada para verificar performance no conjunto de testes.
- ▶ Queríamos observar a transformação da imagem de Input após cada camada da ConvNet
- ▶ A fim de analisar com mais detalhes, foram adicionadas mais camadas convolucionais no modelo apresentado anteriormente.

# Apresentacao do Fábio - Sugestão: Introdução a exploração dos resultados

# Apresentacao do Fábio - Sugestao: Apresentacao das imagens dos filtros e saidas das convolucoes

# Apresentacao do Fábio - Sugestão: Discussao sobre como essas observacoes ligam na MLP clássica e/ou problemas de Machine Learning